Distributed Admission Control via Dual-Queue Management

Dongsook Kim and Mingyan Liu Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI
E-mail: kimds,mingyan@eecs.umich.edu

***Abstract*— This paper introduces and studies a new queue management framework via the use of connection level information already embedded in existing data traffic. Our goal is to improve the system performance and resource utilization at times of intense network congestion. Under this framework, data packets and connection establishment packets are queued separately. Different queueing and dropping policies can then be applied to them, thus the term *dual-queue* management. This framework is stateless and does not require per-flow queueing or flow counting. Using examples of such policies our initial simulation shows that this scheme can lead to much higher performance and network resource utilization during severe congestion. It also leads to more robust and scalable network design. Moreover, it provides an attractive mechanism for network provisioning via parameter tuning.**

I. Introduction

Much of today's data traffic is *best-effort* based (e.g., TCP or UDP). Although best-effort does not provide any performance guarantee, most of the applications using best effort-service do require certain minimal performance to be of practical use. Examples include voice over IP where packet losses above a certain level can completely disrupt a conversation. Web browsing is another example where prolonged delay in downloading a page can cause the user to abort the application and walk away. At times of high congestion, not only does end users experience very poor performance, but a lot of network resource is wasted due to retransmissions and aborted connections.

The regulation of best-effort traffic and congestion control is largely left to queue management. FIFO drop-tail (DT) and Random Early Discard (RED) based queueing are widely deployed. There has been extensive research in active queue management (AQM) [1], [3]. These methods focus on dropping or marking within a queue to regulate flow rates. They can be quite effective in reducing packet losses under certain level of congestion. In particular, method introduced in [3] decouples congestion measure from performance measure and results in much higher resource utilization and lower packet losses. However, the user-experienced performance is ultimately determined by demand and supply. At times of intense user demand, even the ideal congestion control cannot provide the desired level of performance within the context of best-effort. In contrast to best-effort service, guaranteed service provide QoS guarantees via resource reservation and per-flow queueing. Traffic is typically classified into separate flows and reservation-based scheduling is performed among different flows (e.g., Diffserv and RSVP). Per-flow queueing requires expensive and complicated hardware and software, and is hard to be made scalable in general. Consequently, applications will continue to be developed that use best effort service, while their usefulness only exists given certain minimum quality guarantee (deterministic or probabilistic). This has motivated us to look for alternative inexpensive solutions for improving the performance of best effort services in a stochastic sense.

Since applications are usually connection-oriented, either at the application layer or at the transport layer or both, the key to our approach is to utilize such connection level information embedded in the traffic, usually via certain field in the packet header. Our queueing framework consists of separate queues for different types of *packets* (which may belong to the same flow) instead of separate queues for different types of *flows* (which may contain different types of packets). This does not require the router to keep any per-connection or per-flow states, nor does it interfere with functionalities of end-to-end protocols. Current router technologies on fast look-up can easily realize this scheme. Once we place connection establishment packets and data packets into separate queues different drop policies can be applied. We will thus call this scheme the Dual-Queue Management scheme (DQM). By adopting a more aggressive policy to the connection establishment packets in the event of heavy congestion, e.g., delay or drop more connection establishment packets, we can maintain a small enough drop/mark probability for data packets, so that existing connections can go through and complete rather quickly and release network resources. This scheme can be seen as a generalized distributed admission control mechanism, as well as an enhanced AQM mechanism. It may prove especially useful in a wireless network where resources are highly constrained.

Although this framework can be applied to various applications and services, in the rest of this paper we will focus on TCP as an example to show the details of this approach. In this case two separate queues are maintained, one for SYN and SYN/ACK packets and one for the rest. We will ignore whether these two queues are virtual queues or physical queues since it does not affect our illustration of the approach. We will assume that each queue has been provisioned to have a fixed queue size. Admission control for TCP has been studied in the literature. [4] thoroughly discussed congestion collapse and inefficiency of the network mathematically, and advocated the use of admission control for TCP connections in order to ensure a minimum acceptable throughput. A closely related

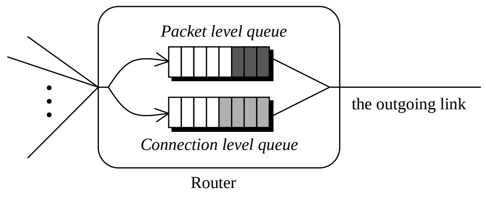


Fig. 1. *The system model for DQM.*

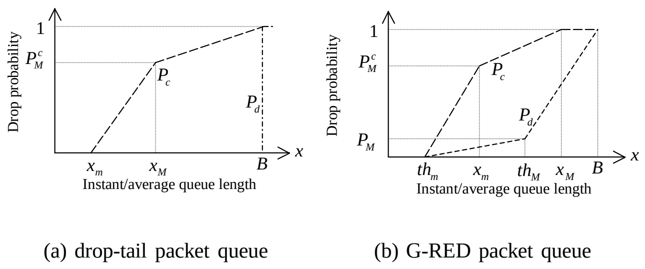


Fig. 2. *Design parameters for connection queue.*

paper is [5] where an implicit admission control scheme was proposed by deferring SYN packets at the routers. The decision to interfere with the establishment of a new connection is based on the load estimator at the routers using the notion of effective bandwidth. Our work does not rely on such estimate. [6] proposed an admission control mechanism activated by detecting extreme overload condition, which requires per-session state at routers. In contrast, our scheme does not require any flow state information.

The rest of the paper is organized as follows. We present the details of DQM in Section II along with discussions on the features and limitations of this scheme. In Section III, we present simulation results obtained using NS-2. We conclude with future work in Section IV.

II. The Dual-Queue Management

The basic concept underlying our approach is to separate data packets and connection establishment packets into two separate queues, shown in Figure 1. More specifically, in the case of TCP, each router keeps one queue for SYN and SYN/ACK packets and another for normal data packets. We will call the SYN and SYN/ACK packet queue the *connection queue* and the other queue the *packet queue*. Although connection dropping occurs in pure drop-tail and RED along with data packet dropping, it typically is far from sufficient to provide good per-connection performance where there are a large amount of incoming connections. The idea of our scheme is to introduce higher drop probability for SYN packets in order to reduce the number of connections in the network at times of congestion. Thus a smaller number of connections will achieve higher individual throughput and lower latency at the expense of a higher blocking probability. By doing so existing connections release resources sooner, and the utilization of resources can be improved.

We illustrate one example of possible realizations of such a scheme using a linear relationship between packet queue length and the connection drop probability, as shown in Figure 2. Figure 2(a) shows the connection drop probability as a function of a drop-tail packet level queue length. Here B is the packet queue size and P_d is the data packet drop probability. P_c is the connection drop probability (i.e., blocking probability), which is determined by two thresholds x_m and x_M. P_M^c is the drop probability at the higher threshold (or the turning point of the two linear relations). Within this family, a drop policy is completely determined by the triple (x_m, x_M, P_M^c). Figure 2(b) shows the connection drop probability as a function of a G-RED (gentle mode) packet queue length. th_m and th_M are two thresholds of G-RED. This family of policies can be completely determined by the same triple given a set of G-RED parameters. In these illustrations, x can be either instantaneous queue length or average queue length. The above policy characterization does not reflect the scheduling between two queues. In reality not only can we drop, but we can also delay connection packets. Our current simulation studies a simplified policy where the connection packets and data packets are transmitted on a FIFO basis, meaning that the connection packets are only dropped but not delayed.

An attractive feature of DQM is its simplicity. As mentioned before, DQM in essence is a distributed admission control mechanism, but realized by managing two isolated queues with different but correlated dropping parameters. It can simply be viewed as an enhancement to current queuing schemes (TD or RED). However, this simplicity is not without a price. Since we do not keep flow information, it can be hard to infer the amount of active flows in the system simply based on the packet queue size. Therefore how to properly correlate the parameters of these two queues can be a difficult task. A thorough treatment of the subject is out of the scope of this paper. In the next section we will focus on comparing the performance of DQM with pure RED or DT under a few sets of representative parameters (with varying degree of aggressiveness).

III. Simulation Results

We performed a set of simulation using NS-2 simulator [7]. Simulated topologies include a single bottleneck scenario and a multiple router scenario. We used the `full-tcp` agent with Reno for each connection with a packet size of $L = 576$ bytes. If a SYN packet is dropped at the router, each connection retries after timeout as specified in `full-tcp`. We examine the router performance in the cases of drop-tail (DT), gentle-RED (G-RED) [8], and DQM connection drop policies defined by $\{x_m,\ x_M,\ P_M^c\}$. The connection drop probability is a function of the average packet queue length. The performance metrics we measured are summarized as follows:

- *The average throughput and latency per flow* (T,D): These are calculated for flows that are admitted.
- *The average number of admitted connections present in the system* (N).
- *The number of flows completed per second.*

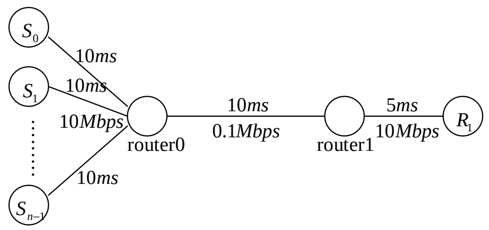


Fig. 3. Simulation model for a single bottleneck link.

	Drop-tail	G-R1	G-R2	G-R3
$T(bps)$	890.8	1527.7	1126.1	1299.5
$D(sec)$	196.6	96.5	174.1	172.1
P_d	0.48	0.26	0.4	0.45
P_c	0.30	0.77	0.6	0.57

TABLE I

A MODERATELY CONGESTED BOTTLENECK LINK WITH PURE DROP-TAIL AND G-RED ONLY.

- *The number of retransmitted packets*: This is the total number of retransmitted packets among all flows over a fixed period of simulation duration.
- *Average queue length.*
- *The average drop probabilities of connection queue and packet queue* (P_c,P_d).

A. A single bottleneck link

In this set of simulation, we consider a single bottleneck scenario depicted in Figure 3. The capacity of the bottleneck link is $C = 0.1$ Mbps and "router0" has a queue size of 90 packets. TCP flows generated by sender nodes (S_n) arrive at the router as a Poisson process. The size of flows are exponentially distributed.

1) Homogeneous flows: We first consider homogeneous flows with average flow size $d = 20$ packets and two way propagation delay 50 ms, over a simulation duration of 1500 seconds. We will vary the offered load on the system, denoted by ρ, expressed as $\rho = \frac{\lambda \cdot d \cdot L}{C}$. This is achieved by varying accordingly the arrival rate of TCP flows, denoted by λ. We will first show the performance using TD and RED under three sets of parameters and then compare some of them with DQM under three sets of parameters.

Table I contains a few measures under drop-tail and G-RED, when ρ is set to a moderate 1.2. The three sets of G-RED parameters are: $\{th_m,\ th_M,\ P_M\}$={15, 60, 0.1}(G-R1), {15, 45, 0.1}(G-R2), and {15, 30, 0.4}(G-R3). We see that as the dropping parameters become more aggressive, packet dropping increases. Interestingly however, the connection dropping decreases in the case of G-RED. In all these case data packets experience very high drop probabilities, which inevitably leads to retransmissions.

Consider now the following three DQM parameters, which we will use together with G-R2 for performance comparison: $\{x_m,\ x_M,\ P^c_M\}$={45, 90, 0.4} (P1), {45, 90, 0.7} (P2) and {45, 45, 1} (P3), depicted in Figure 4 (where G-RED refers to G-R2).

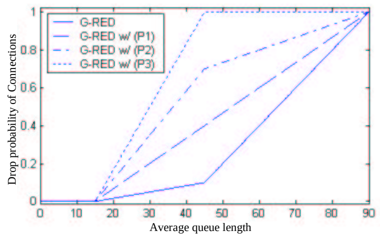
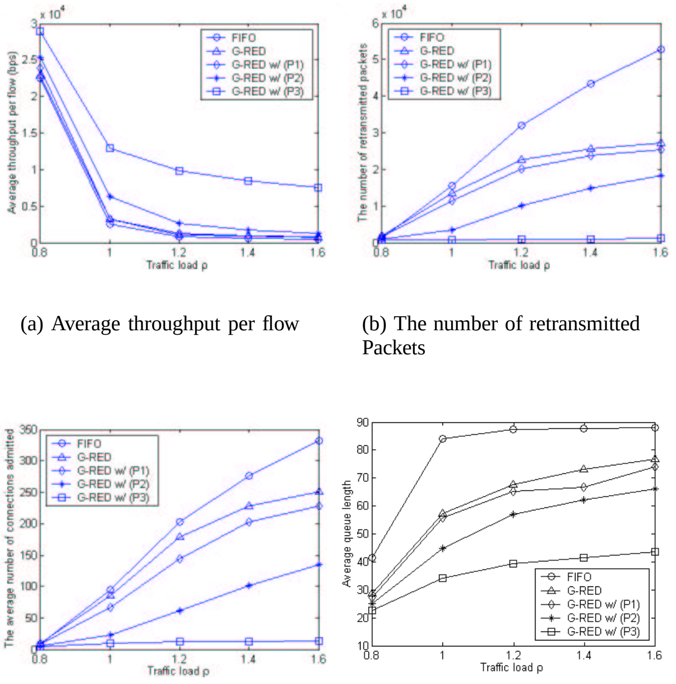


Fig. 4. Drop policies used in the simulation.



(a) Average throughput per flow

(b) The number of retransmitted Packets

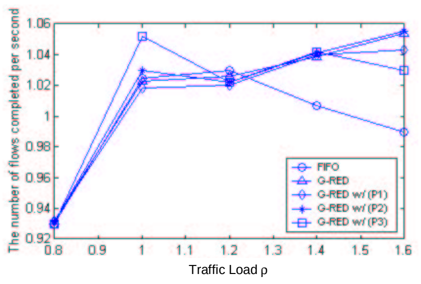
(c) Average number of admitted flows in the link

(d) Average queue length

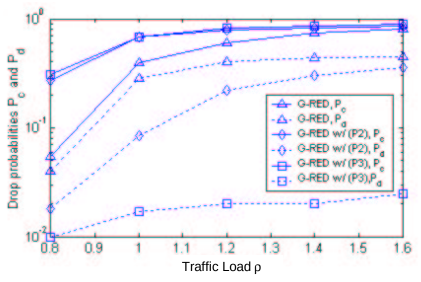
Fig. 5. Simulation results without and with DQM on a single bottleneck link for homogeneous traffic.

Figure 5 shows results comparing the three DQM policies with G-RED as well as simply DT. These results are to a large extent to be expected. P1 has a fairly similar performance as G-RED (G-R2), while P3 performs significantly different from all others as ρ increases. It is able to maintain a reasonable level of per connection throughput, and an extremely low packet retransmission count, by rejecting a large number of incoming connection. In the case of DT, the amount of retransmission grows linearly with ρ, while G-RED performs better. This is consistent with earlier results shown in Table I, in that G-RED is able to drop a large amount of connection requests while DT is not. P1 and P2, which are less aggressive than P3, perform in between G-RED and P3.

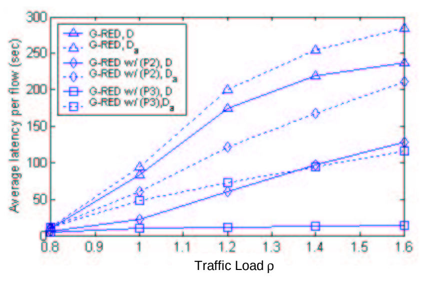
Figure 6(a) shows the rate of connection completion as a measure of link utilization under different schemes. Initially when the load is low (below 1.0), connection completion rate grows as the load due to link under-utilization. For DT, the rate starts going down as ρ increases beyond 1.2. This indicates that



(a) The number of flows completed per second



(b) Drop probabilities of connections and data packets



(c) Average latency per flow

Fig. 6. Simulation results G-RED and DQM P2 and P3 for a single bottleneck link with homogeneous traffic.

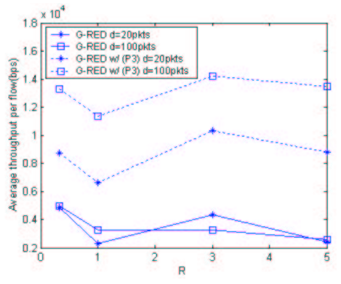
most of the resources are wasted in retransmission, so even though there are more connections coming in, less is going out. G-RED on the other hand is able to maintain a slight increase, indicating that early dropping is effective to keep up with the load. P2 performs very closely to G-RED, while P1 and P3 exhibit slight fluctuation. Overall, DQM with the given parameters is able to maintain as high a link utilization as G-RED, even though large amount of connections are dropped. In particular, P3 has the highest completion rate at $\rho = 1$, indicating that connections are being served faster and release the resources faster. Combining results in Figure 6(a) with that in Figure 5 we see that DQM achieves similar level of link utilization but but much better per connection performance. It's worth pointing out that the high utilization of G-RED is due to the large amount of connections present in the system (see Figure 5(c), while the high utilization of DQM is due to the much smaller amount of connection present in the system that releases resources much faster so as to allow more new connections to be admitted. It is because of this DQM results in much less resource wastage (retransmissions) and much higher per connection performance.

Figures 6(b) and (c) compare the connection and data packet drop probability, as well as connection latency under G-RED, P2 and P3, respectively. In 6(c), two types of latencies are defined. D refers to the average latency a connection experiences *after* it has been admitted, i.e., D does not include the delay in transmitting SYN and getting SYNACK back. D_a refers to the average total latency *including* the connection setup phase where a connection request may timeout and be retried. At $\rho = 1.6$, the average latency D under P3 is 13.96 sec, which is about 5.9% of the latency under $G-RED$, while the total latency D_a under P3 is till 40.7% of that under G-RED. As ρ increases, the difference between latency and total latency becomes larger. This is because as the link gets more congested, more SYN packets are dropped. These connections then timeout and retry after backoff, thus the total latency increases at a higher rate.

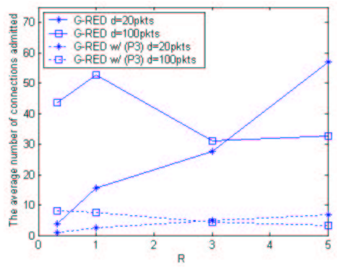
All above results show that with the DQM framework, we are able to maintain high link utilization while improving individual connection's performance via admission control. With this framework we have a lot of flexibility in determining the performance of individual connection via parameter setting. It also presents an attractive approach to network provisioning and capacity planning.

2) *Heterogeneous flows:* In this subsection we will examine one type of heterogeneity, which is flows of different sizes. Among others, this is probably the most critical and interesting. Due to the greedy nature of TCP, a small number of long flows may occupy a large amount of packet queue space. Ideally we would like to exercise admission control solely based on the amount of flows in the system and not their sizes. However, since DQM does not keep states or count the number of flow admitted/present, it does not distinguish between a large packet queue caused by numerous small flows and a large queue caused by a few large flows. As we show below, this does lead to unfairness between large and short flows. However, all the benefit as we pointed out before are maintained, including low packet drop probability and packet retransmission, higher per connection throughput, and link utilization.

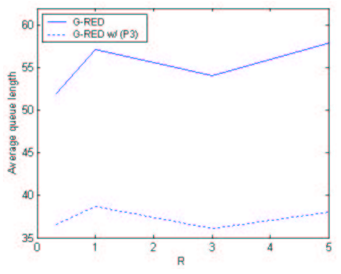
Using the same single bottle network topology we simulated two class of flows, with average sizes of 20 packets and 100 packets, respectively. They have the same 50 ms propagation delay. We denote by R the ratio between the arrival rates of the class of short flows and the class of long flows. Figure 7 shows the performance comparison under G-RED and P3 by varying R and fixing $\rho = 2$. We see from Figure 7(a) that DQM leads to an obvious performance difference between the short and long flows, with much higher throughput for the long flows. In contrast, G-RED exhibits a much better fairness, although both classes of flows experienced much lower throughput than that under DQM. From Figure 7(b), both drop schemes result in similar patterns of admitted flows, though on a very different scale. Again packet drop probability is much lower under



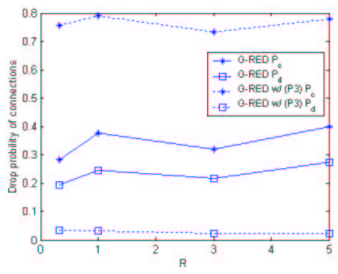
(a) Average throughput per flow



(b) Average number of admitted flows in the link



(c) Average queue length



(d) Drop probability of connections

Fig. 7. Two classes of flows.

DQM, resulting in fewer number of retransmissions.

We also conducted simulation under light traffic load, e.g., for $\rho = 0.6$ or 0.7 under the above scenario. It should be mentioned that the difference between G-RED and DQM P3 in this case is very limited, even with the presence of very long flows. In general, DQM results in a slightly lower queue occupancy and number of admitted connections, and a slightly higher connection drop probability.

B. Multiple congested links

We have argued that due to much reduced amount of packet retransmission, DQM puts network resource to more efficient use. This can probably more clearly demonstrated using a multiple congested link example. This is because when the network is highly congested and in multiple places, a packet being dropped at a downstream router renders all upstream transmissions wasted. With DQM this is more likely to happen to connection set up packets, and much less to the data packets. We consider a tandem network composed of three routers depicted in Figure 8. Three classes of TCP flows are generated based on their source-destination pairs. The arrival rate and the average size of flows are the same for all classes. Average size of a flow is 20 packets. The traffic load on each link is $\rho_i = \frac{2 \cdot \lambda \cdot d \cdot L}{C}, i = 1, 2$. We set $\rho_1 = \rho_2 = 1.2$ via adjusting λ.

Table II compares the amount of packet retransmissions experienced by class 3 flows with G-R2 (in all routers) vs. with P3 (in all routers) over a simulation duration of 700 seconds. We see that the total number of retransmitted packets is much lower with P3, more significant than the single router/link case.

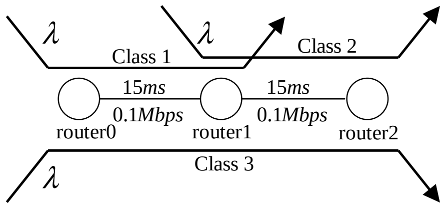


Fig. 8. Simulation model for multiple routers.

	G-RED	P3
Latency D (sec)	118.80	27.63
Retransmitted packets	3364	127

TABLE II
CLASS 3 IN A MULTIPLE-ROUTER SCENARIO.

This is precisely because more aggressive connection dropping reduces the chance of a packet being dropped at a downstream router.

IV. CONCLUSION

In this paper, we developed and studied a new approach to traffic management in the Internet using a dual-queue management framework by separating traffic into a data packet queue and a connection queue and dropping connection establishment packets more aggressively. The two queues are separate but have correlated dropping parameters. This is both a distributed admission control scheme as well as an enhanced AQM scheme. It does not rely on state information at the routers and can be very easily implemented using current technologies. We showed an example of such connection drop policies and presented simulation results for both a single-bottleneck and a multiple bottleneck scenario. These results show that the performance experience by the active connections are greatly improved at the expense of increased connection blocked. However, due to the much reduced number of retransmissions, the utilization of the network remains high.

REFERENCES

[1] C.V. Hollot, V. Misra, D. Towsley, and W. Gong, “A Control Theoretic Analysis of RED,” *Proceedings of the 2001 IEEE Infocom*, vol. 3, 2001.

[2] C.V. Hollot, V. Misra, D. Towsley, and W. Gong, “On Designing Improved Controllers for AQM Routers Supporting TCP Flows,” *Proceedings of the 2001 IEEE Infocom*, vol. 3, 2001.

[3] D. Lapsley and S. Low, “Random Early Marking for Internet Congestion Control,” *Proceedings of the 1999 IEEE Globecom*, vol. 3, 1999.

[4] L. Massoulie and J. Roberts, “Arguments in favour of admission control for TCP flows,” *Proceedings of ITC-16: Teletraffic Engineering in a Competitive World*, Jun. 1999.

[5] R. Mortier, I. Pratt, C. Clark, and S. Crosby, “Implicit Admission Control” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, Dec. 2000.

[6] H. Jiang and C. Dovrolis, “Guardian: A Router Mechanism for Extreme Overload Prevention,” *Proceedings of Scalability and Traffic Control in IP Networks*, Aug. 2002.

[7] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns.

[8] S. Floyd, Recommendations on using the gentle_ variant of RED, Mar. 2000. http://www.aciri.org/floyd/red/gentle.html.